University of Houston

Homework 3

COSC 6377

Computer Networks

Omprakash Gnawali

NAME: Arti Patel

UHID: 2329771

Due: Thursday, November 16, 2023

11:59 PM

Objective:

The main objective of this assignment is to investigate and understand the delay introduced by a lazy HTTP server in response to POST requests with varying "truncid" values. The server introduces delays that are influenced by the provided "truncid," and we need to analyze the delay function implemented by the server.

Program development:

1. Developed a Python program to send POST requests to the web application's "/delayme" endpoint, simulating a form submission with a "truncid" field
2. Ensure that the requests are correctly formatted and return a JSON response with a "status" attribute indicating either "GOOD" or "BAD."
3. Added a code to record multiple latency data with both endpoints (with delay and without delay)

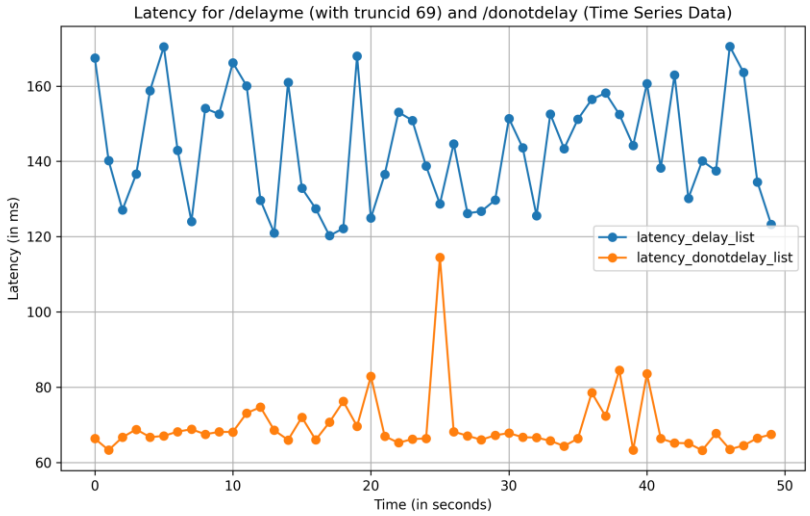
Data Collection:

1. The program has a feature to record the time between the request and response for multiple measurements. The recorded delays were stored for further analysis.

Results:

1. Analysis of Delay Patterns:
    a. Analyzed the collected delay data patterns or correlations with "truncid" values.
    b. Observed the delay patterns between two endpoints "/delayme" and "/donotdelayme" endpoints.
    c. Here are few screenshots of with and without delay endpoints with different “truncid” and their statistical analysis

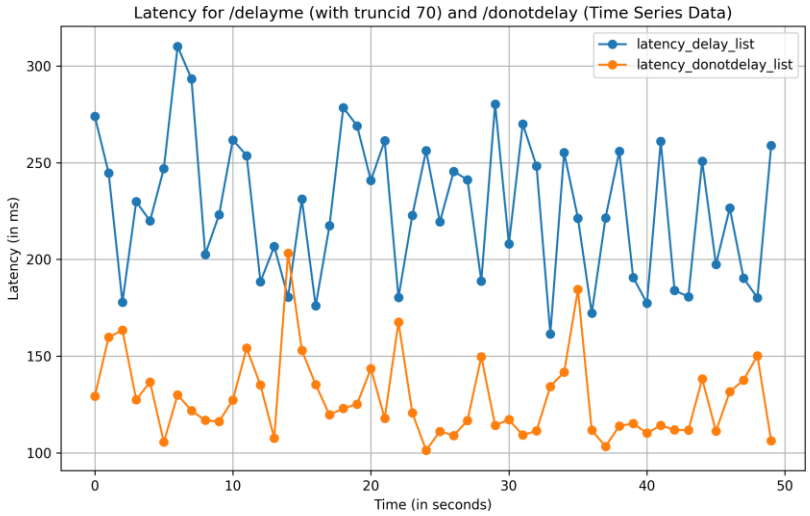
d. For “truncid” = 69




```
Statistical Summary for not_delay_measurements:
         count       mean       std        min        25%        50%        75%         max
truncid
69        50.0  69.514146  8.157928  63.201904  66.061735  67.058206  68.790495  114.494324

Statistical Summary for delay_measurements:
         count        mean        std         min         25%         50%         75%         max
truncid
69        50.0  143.667746  15.177279  120.246649  129.669547  143.172383  155.85804  170.531034
```

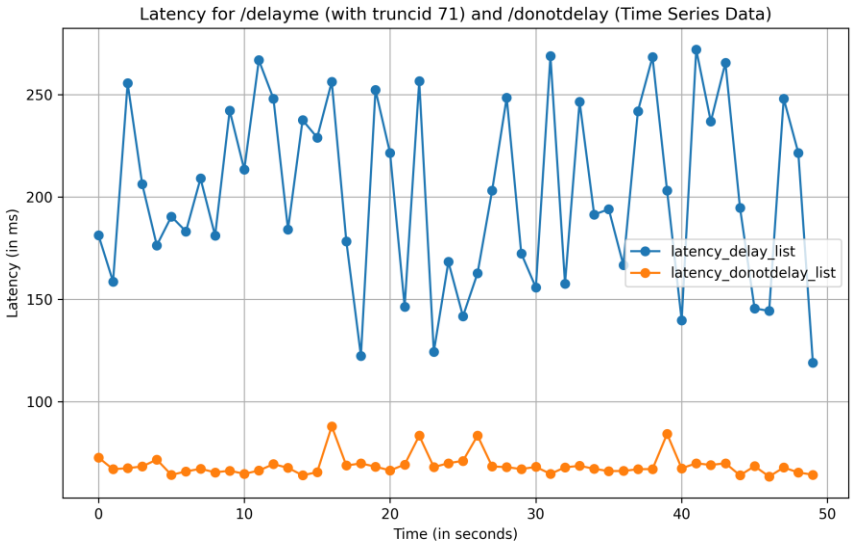
e. For “truncid” = 70:




```
Statistical Summary for not_delay_measurements:
         count        mean        std         min         25%         50%         75%         max
truncid
70        50.0  128.201165  21.537157  101.367235  111.800492  121.274352  137.427628  203.271151

Statistical Summary for delay_measurements:
         count       mean        std         min         25%         50%         75%         max
truncid
70        50.0  226.67726  36.779533  161.434889  190.408289  224.783659  255.759239  310.056925
```

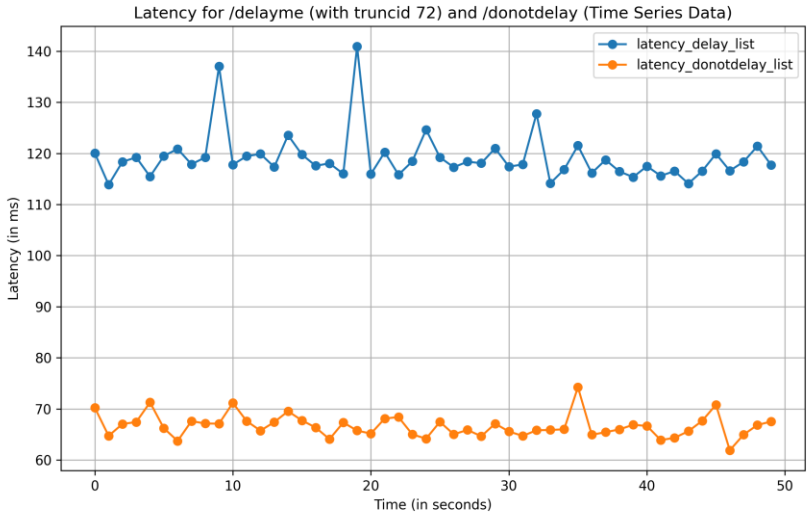
f. For ”truncid” = 70




```
Statistical Summary for not_delay_measurements:
         count       mean       std        min        25%        50%        75%        max
truncid
71        50.0  68.801737  5.179906  63.397408  66.162467  67.741871  69.201827  87.949991

Statistical Summary for delay_measurements:
         count        mean        std         min         25%         50%         75%         max
truncid
71        50.0  201.938562  45.018787  118.994236  167.105079  198.870182  245.409429  271.930933
```

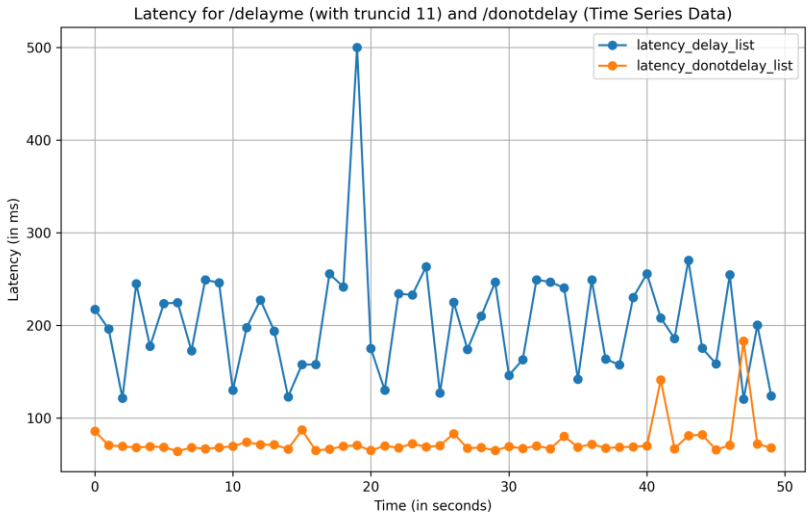
g. For truncid = 72




```
Statistical Summary for not_delay_measurements:
         count       mean       std        min     25%        50%        75%       max
truncid
72        50.0  66.643662  2.219479  61.894655  65.063  66.289544  67.519546  74.25189

Statistical Summary for delay_measurements:
         count        mean       std         min         25%         50%         75%         max
truncid
72        50.0  119.145308  4.861617  113.869905  116.552353  118.061543  119.859397  140.923262
```

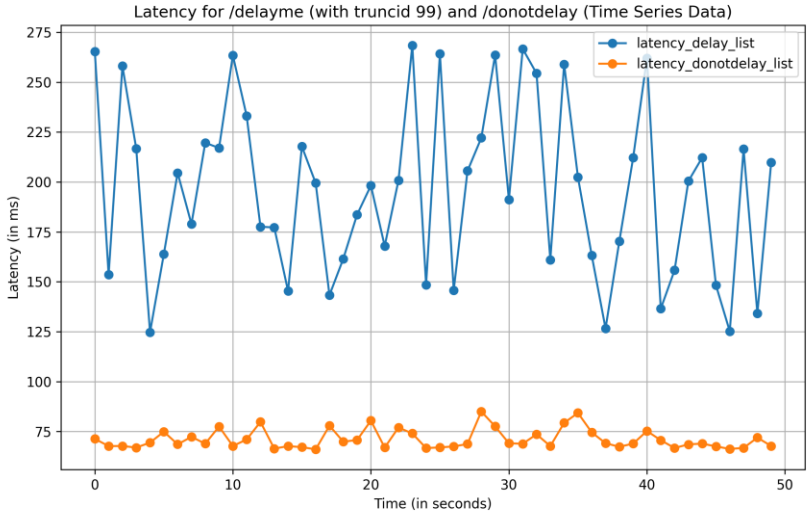
h. For “truncid” = 11




```
Statistical Summary for not_delay_measurements:
         count       mean        std        min       25%       50%       75%        max
truncid
11        50.0  73.959832  19.345087  63.844204  67.50077  69.031477  71.046472  182.89423

Statistical Summary for delay_measurements:
         count        mean        std         min         25%         50%         75%    max
truncid
11        50.0  204.219465  62.505243  120.132208  159.610987  204.047322  244.003475  500.0
```

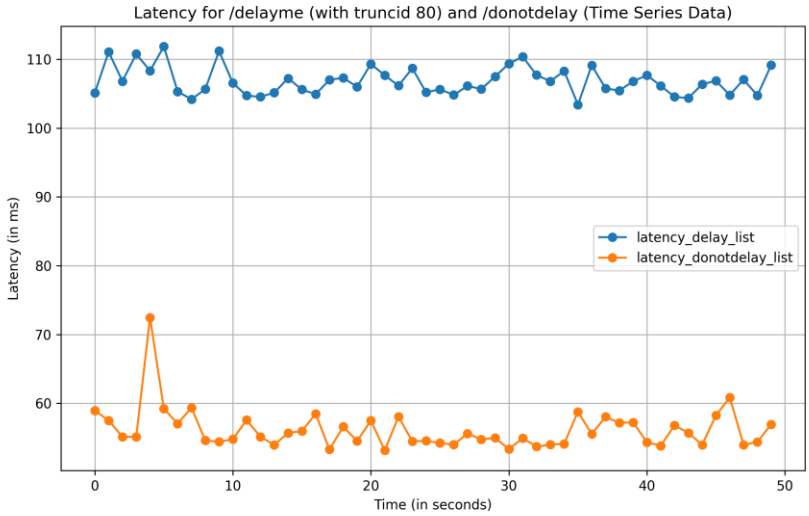
i. For “truncid” = 99




```
Statistical Summary for not_delay_measurements:
         count       mean       std        min       25%       50%       75%        max
truncid
99        50.0  71.076961  4.899325  66.066027  67.557156  68.93456  73.914826  85.016012

Statistical Summary for delay_measurements:
         count        mean        std         min         25%         50%         75%         max
truncid
99        50.0  195.961599  44.039496  124.736786  161.122024  199.941516  219.179094  268.417835
```

j. For “truncid" = 80 (tried on google Collaboratory)




```
Statistical Summary for not_delay_measurements:
         count       mean       std        min        25%        50%  \
truncid
80        50.0  56.125498  3.024422  53.151369  54.324329  55.116177

                75%        max
truncid
80        57.403266  72.456598

Statistical Summary for delay_measurements:
         count       mean       std         min         25%         50%  \
truncid
80        50.0  106.82766  2.034599  103.404284  105.232179  106.473446

                75%         max
truncid
80        107.721388  111.877918
```

k. The presence of intentional delays (like we can see the /delayme endpoint graph is using truncid as delay time) introduces variability in response times. on the other hand, the latency observed without intentional delays (like /donotdelayme endpoint) represents the baseline response time of the server. We can see from the graph that there is some significant difference in response times. indicates the impact of intentional delays on overall system latency.

l. We observed no direct correlation between the latency I`ntroduced by intentional delays and the baseline latency without delays. The variations in response times under intentional delays did not exhibit any consistent or noticeable pattern when compared to the latency observed without intentional delays. This lack of correlation suggests that the intentional delays, as introduced by the server based on the 'truncid' parameter, may not exhibit a straightforward relationship with the overall system latency. The baseline latency without delays appears to be

independent of the intentional delays, indicating that other factors or mechanisms might be influencing the observed variations in response times.

m. However, in detailed analysis I found intriguing patterns in latency that distinguish between intentional delays and baseline latency without delays. Interestingly, latency under intentional delays exhibits a notable stability, particularly when the 'truncid' parameter involves even numbers (that is half of truncid is even number). Specifically, latency values are consistently more stable when the 'truncid's half is an even number, showcasing a discernible pattern. (Example: when truncid is 72(half is 36) the graph is more stable, same for truncid 80 (half is 40)

Python code:

```
import requests
import time
from datetime import datetime
import csv
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# 1. Setting parameters for the POST request
url = 'http://35.224.82.70:5555/donotdelayme'
url1 = 'http://35.224.82.70:5555/delayme'
# headers = {'Content-type': 'application/json'}
data = {"truncid": 72}

latency_donotdelay_list = []
latency_delay_list = []

not_delay_measurements = []
delay_measurements =[]

for i in range(50):
    # 2. Sending the POST request
    start_time = time.time()
    response = requests.post(url,data=data)
    latency = (time.time() - start_time)*1000
    latency_donotdelay_list.append(latency)
    not_delay_measurements.append({"truncid": data['truncid'], "delay":
latency, "status": response.json()["status"]})

    if response.status_code == requests.codes.ok:
```

```
        print('Request was successful for:',url)
        print(response.json())
        print(f"Latency: {latency} ms")
    else:
        latency = (time.time() - start_time)*1000
        print('Request failed for:',url,' with status code:',
response.status_code)
        print(f"Latency: {latency} ms")
    time.sleep(1)
    start_time = time.time()
    response = requests.post(url1,data=data)
    latency = (time.time() - start_time)*1000
    latency_delay_list.append(latency)
    delay_measurements.append({"truncid": data['truncid'], "delay": latency,
"status": response.json()["status"]})

    if response.status_code == requests.codes.ok:
        print('Request was successful for:',url1)
        print(response.json())
        print(f"Latency: {latency} ms")
    else:
        latency = (time.time() - start_time)*1000
        print('Request failed for:',url1,' with status code:',
response.status_code)
        print(f"Latency: {latency} ms")

    time.sleep(2)

def analyze_data(measurements):
    df = pd.DataFrame(measurements)

    # Statistical summary
    summary = df.groupby('truncid')['delay'].describe()
    return summary

# print(latency_delay_list)
# print(latency_donotdelay_list)

not_delay_measurements_summary = analyze_data(not_delay_measurements)
print(f"\nStatistical Summary for not_delay_measurements:")
print(not_delay_measurements_summary)

delay_measurements_summary = analyze_data(delay_measurements)
print(f"\nStatistical Summary for delay_measurements:")
print(delay_measurements_summary)
```

```
# Create a DataFrame
df = pd.DataFrame({'delay': np.array(latency_delay_list), 'notdelay':
np.array(latency_donotdelay_list)})

# Plotting
plt.style.use('default')
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['delay'], label='latency_delay_list', marker='o')
plt.plot(df.index, df['notdelay'], label='latency_donotdelay_list', marker='o')

plt.title(f'Latency for /delayme (with truncid {data["truncid"]}) and
/donotdelay (Time Series Data)')
plt.xlabel("Time (in seconds)")
plt.ylabel('Latency (in ms)')
plt.legend()
plt.grid(True)
plt.savefig('latency_plot.png',dpi=300)
```